Single-cell RNA sequencing worksheet Day7 part 2 (Project A)

Useful resource: https://satijalab.org/seurat/articles/seurat5_integration

Worksheet overview:

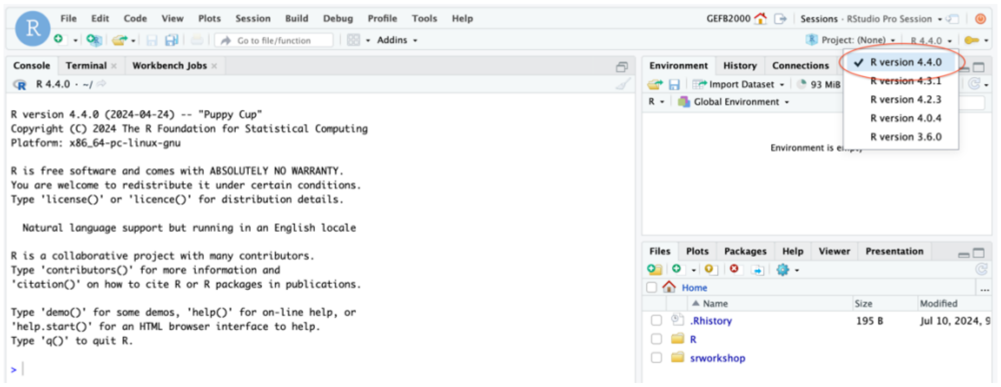
Prior to this worksheet, fastq files were mapped using the *Cellranger* suite. The 'cellranger count' command generates several output files organized into a series of directories. Within the '../outs/' directory is a directory called '../outs/filtered_features_bc_matrix/'. In this directory are 3x tsv files: matrix.mtx.gz, barcodes.tsv.gz, and features.tsv.gz. You will provide this 'filtered_features_bc_matrix' directory using the *Seurat* package in *R* to read in these files. The ensuing Seurat object contains, among much else, a counts matrix where each row is a gene and each column corresponds to a single cell (depicted as the UMI barcode – so each single cell appears as a string of nucleotides which correspond to the UMI that cell was labeled with). The values comprising this count table are the number of reads (from the fastq file) that were mapped (using *Cellranger*) to the corresponding gene in the row. We then go through a series of cleaning, normalizing, and scaling these *Seurat* objects individually prior to integrating all samples together (covered in next worksheet).

Covered in this worksheet:

1) Setting *R* version and loading Seurat package into R environment
2) Reading files into R (Seurat) that were mapped using Cellranger
3) Creating Seurat objects and adding metadata
4) Filtering out low-quality cells and doublets
5) Merging Seurat objects
6) Normalizing and scaling data, and identifying highly variable genes across single cells
7) Conducting principal component reduction (PCA)
8) Integrating Seurat objects by gender differences

Code:

1) Setting *R* version and reading files into R (Seurat) that were mapped using Cellranger


File Edit Code View Plots Session Build Debug Profile Tools Help
GEFB2000
Sessions · RStudio Pro Session
Go to file/function
Addins
Project: (None)
R 4.4.0
Console Terminal Workbench Jobs
R 4.4.0 · ~/
Environment History Connections
Import Dataset
93 MiB
Global Environment
✓ R version 4.4.0
R version 4.3.1
R version 4.2.3
R version 4.0.4
R version 3.6.0
Files Plots Packages Help Viewer Presentation
Home
Name Size Modified
.Rhistory 195 B Jul 10, 2024, 9
R
srworkshop

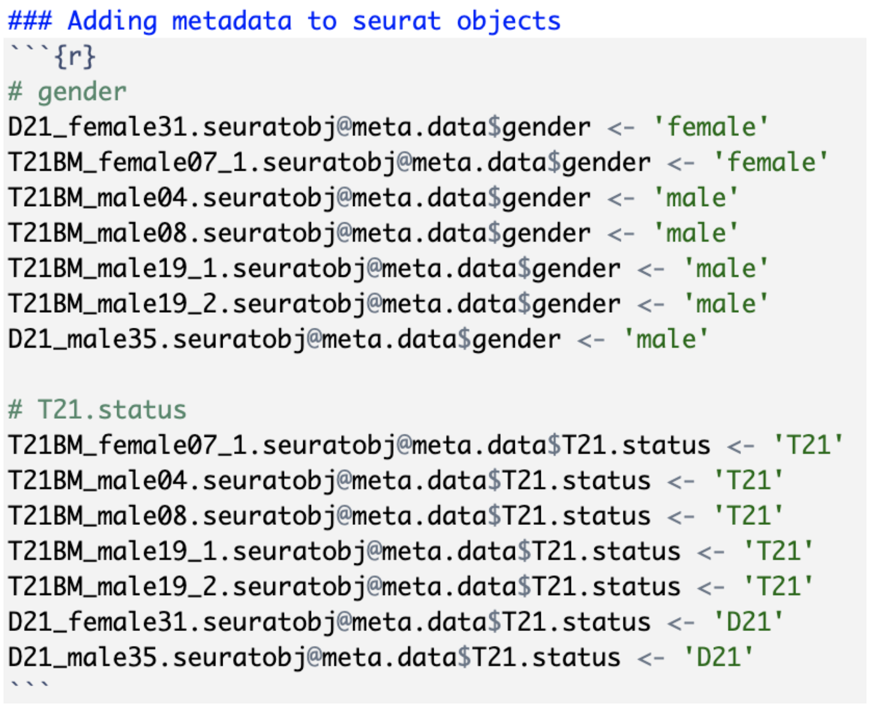
```
R version 4.4.0 (2024-04-24) -- "Puppy Cup"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

~~~
```{r}
library(Seurat)
library(tidyr)
library(dplyr)
```
~~~

2) Reading files into R (Seurat) that were mapped using Cellranger and creating Seurat objects. The directory we're looking for is called 'filtered_feature_bc_matrix' which is in the "/outs/" directory that is produced by *Cellranger*. Make sure to direct the Read10x() function to the correct 'filtered_feature_bc_matrix'.

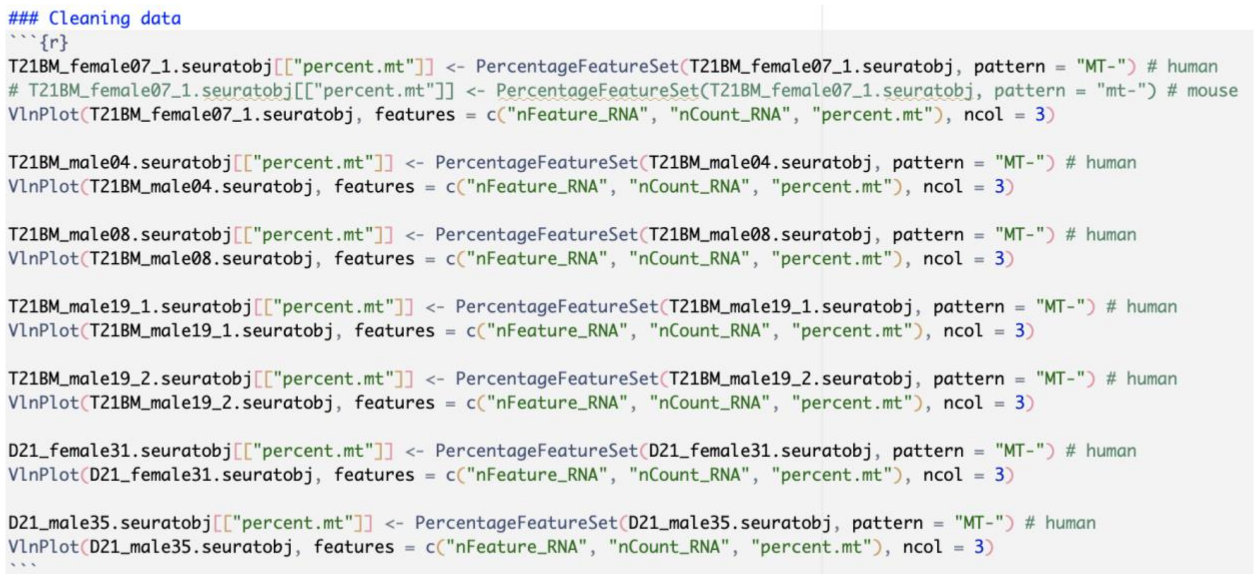
~~~
### Reading in filtered_feature_bc_matrix from the cellranger outs/ directory and making seurat objects
```{r}
# AWS directory
workdir <- "/scratch/Shares/public/sread2024/cookingShow/ProjectA/day7/"

T21BM_female07_1 <- Read10X(data.dir = paste0(workdir, "T21BM_female07_1", "/outs/filtered_feature_bc_matrix"))
T21BM_female07_1.seuratobj <- CreateSeuratObject(counts = T21BM_female07_1, project = "T21BM_female07_1")

T21BM_male04 <- Read10X(data.dir = paste0(workdir, "T21BM_male04", "/outs/filtered_feature_bc_matrix"))
T21BM_male04.seuratobj <- CreateSeuratObject(counts = T21BM_male04, project = "T21BM_male04")

T21BM_male08 <- Read10X(data.dir = paste0(workdir, "T21BM_male08", "/outs/filtered_feature_bc_matrix"))
T21BM_male08.seuratobj <- CreateSeuratObject(counts = T21BM_male08, project = "T21BM_male08")

T21BM_male19_1 <- Read10X(data.dir = paste0(workdir, "T21BM_male19_1", "/outs/filtered_feature_bc_matrix"))
T21BM_male19_1.seuratobj <- CreateSeuratObject(counts = T21BM_male19_1, project = "T21BM_male19_1")

T21BM_male19_2 <- Read10X(data.dir = paste0(workdir, "T21BM_male19_2", "/outs/filtered_feature_bc_matrix"))
T21BM_male19_2.seuratobj <- CreateSeuratObject(counts = T21BM_male19_2, project = "T21BM_male19_2")

D21_female31 <- Read10X(data.dir = paste0(workdir, "D21_female31", "/outs/filtered_feature_bc_matrix"))
D21_female31.seuratobj <- CreateSeuratObject(counts = D21_female31, project = "D21_female31")

D21_male35 <- Read10X(data.dir = paste0(workdir, "D21_male35", "/outs/filtered_feature_bc_matrix"))
D21_male35.seuratobj <- CreateSeuratObject(counts = D21_male35, project = "D21_male35")
```
~~~

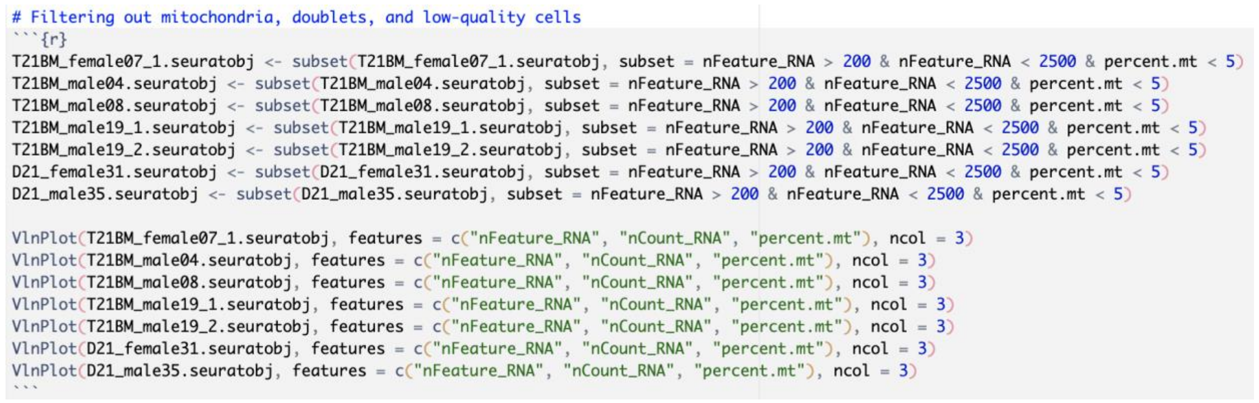
3) Adding metadata to each Seurat object is simple and is done by the “$” just as in a normal *R* object. I have added two metadata designations to each of the Seurat objects (gender and T21.status). In theory, you can add as much metadata as you want depending on the experiment. Metadata becomes important when integrating the data and visualization. (hint: you can never add too much metadata…).



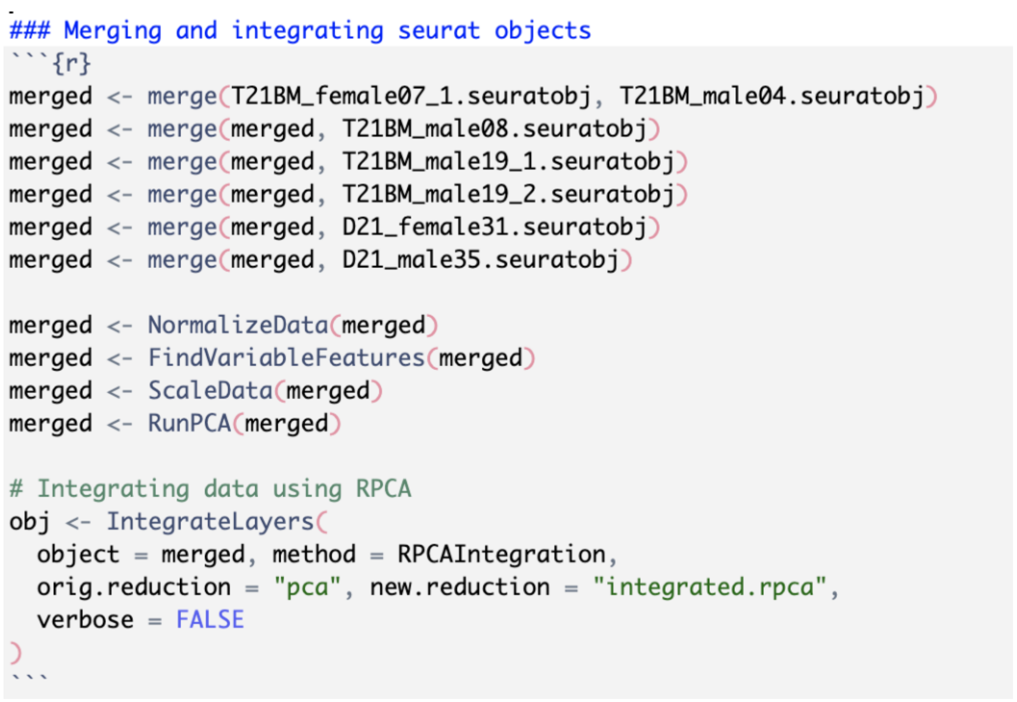
~~~
### Adding metadata to seurat objects
```{r}
# gender
D21_female31.seuratobj@meta.data$gender <- 'female'
T21BM_female07_1.seuratobj@meta.data$gender <- 'female'
T21BM_male04.seuratobj@meta.data$gender <- 'male'
T21BM_male08.seuratobj@meta.data$gender <- 'male'
T21BM_male19_1.seuratobj@meta.data$gender <- 'male'
T21BM_male19_2.seuratobj@meta.data$gender <- 'male'
D21_male35.seuratobj@meta.data$gender <- 'male'

# T21.status
T21BM_female07_1.seuratobj@meta.data$T21.status <- 'T21'
T21BM_male04.seuratobj@meta.data$T21.status <- 'T21'
T21BM_male08.seuratobj@meta.data$T21.status <- 'T21'
T21BM_male19_1.seuratobj@meta.data$T21.status <- 'T21'
T21BM_male19_2.seuratobj@meta.data$T21.status <- 'T21'
D21_female31.seuratobj@meta.data$T21.status <- 'D21'
D21_male35.seuratobj@meta.data$T21.status <- 'D21'
```
~~~

4) Single-cell data can be very messy. To evaluate the quality of our data, we want to visualize:
   a. Distribution of number of mapped genes detected for individual cells (nFeature_RNA)
   b. Distribution of number of reads detected for individual cells (nCount_RNA)
   c. Distribution of number of reads that map to the mitochondrial genome for individual cells (percent.mt) – genes mapped to the mitochondrial genome contain a “MT-“ prefix for human genes, and so we first have to calculate the PercentageFeatureSet() for genes containing this suffix before plotting



~~~
### Cleaning data
```{r}
T21BM_female07_1.seuratobj[["percent.mt"]] <- PercentageFeatureSet(T21BM_female07_1.seuratobj, pattern = "MT-") # human
# T21BM_female07_1.seuratobj[["percent.mt"]] <- PercentageFeatureSet(T21BM_female07_1.seuratobj, pattern = "mt-") # mouse
VlnPlot(T21BM_female07_1.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)

T21BM_male04.seuratobj[["percent.mt"]] <- PercentageFeatureSet(T21BM_male04.seuratobj, pattern = "MT-") # human
VlnPlot(T21BM_male04.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)

T21BM_male08.seuratobj[["percent.mt"]] <- PercentageFeatureSet(T21BM_male08.seuratobj, pattern = "MT-") # human
VlnPlot(T21BM_male08.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)

T21BM_male19_1.seuratobj[["percent.mt"]] <- PercentageFeatureSet(T21BM_male19_1.seuratobj, pattern = "MT-") # human
VlnPlot(T21BM_male19_1.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)

T21BM_male19_2.seuratobj[["percent.mt"]] <- PercentageFeatureSet(T21BM_male19_2.seuratobj, pattern = "MT-") # human
VlnPlot(T21BM_male19_2.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)

D21_female31.seuratobj[["percent.mt"]] <- PercentageFeatureSet(D21_female31.seuratobj, pattern = "MT-") # human
VlnPlot(D21_female31.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)

D21_male35.seuratobj[["percent.mt"]] <- PercentageFeatureSet(D21_male35.seuratobj, pattern = "MT-") # human
VlnPlot(D21_male35.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
~~~
```

Then we filter out

- "Cells" containing < 200 mapped genes which likely aren't actually cells but cell debris or free RNA that was labeled a UMI
- "Cells" with > 2500 mapped genes which likely aren't single cells but are doublets
- "Cells" with > 5% mitochondrial genes, as these are likely mitochondria and not cells

These cutoffs can be modified depending on the distributions observed in the previous step but these values are pretty standard.



```
# Filtering out mitochondria, doublets, and low-quality cells
```{r}
T21BM_female07_1.seuratobj <- subset(T21BM_female07_1.seuratobj, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)
T21BM_male04.seuratobj <- subset(T21BM_male04.seuratobj, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)
T21BM_male08.seuratobj <- subset(T21BM_male08.seuratobj, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)
T21BM_male19_1.seuratobj <- subset(T21BM_male19_1.seuratobj, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)
T21BM_male19_2.seuratobj <- subset(T21BM_male19_2.seuratobj, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)
D21_female31.seuratobj <- subset(D21_female31.seuratobj, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)
D21_male35.seuratobj <- subset(D21_male35.seuratobj, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)

VlnPlot(T21BM_female07_1.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
VlnPlot(T21BM_male04.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
VlnPlot(T21BM_male08.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
VlnPlot(T21BM_male19_1.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
VlnPlot(T21BM_male19_2.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
VlnPlot(D21_female31.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
VlnPlot(D21_male35.seuratobj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```
```

5) Merging Seurat objects (prior to "integration"). First we merge all the Seurat objects and conduct normalization, scaling, and PCA reduction. FindVariableFeatures() identifies genes that are highly variable across cells so Seurat functions can ignore genes that aren't informative in distinguishing different cell types (i.e. housekeeping genes). PCA analysis is the first step towards taking high dimensional data and reducing it to fewer.
```

Then Seurat objects are “integrated” using IntegrateLayers(). There are many different algorithims Seurat offers for integration. In this case I use reciprocal PCA (RPCA) for integration.



~~~
### Merging and integrating seurat objects
```{r}
merged <- merge(T21BM_female07_1.seuratobj, T21BM_male04.seuratobj)
merged <- merge(merged, T21BM_male08.seuratobj)
merged <- merge(merged, T21BM_male19_1.seuratobj)
merged <- merge(merged, T21BM_male19_2.seuratobj)
merged <- merge(merged, D21_female31.seuratobj)
merged <- merge(merged, D21_male35.seuratobj)

merged <- NormalizeData(merged)
merged <- FindVariableFeatures(merged)
merged <- ScaleData(merged)
merged <- RunPCA(merged)

# Integrating data using RPCA
obj <- IntegrateLayers(
  object = merged, method = RPCAIntegration,
  orig.reduction = "pca", new.reduction = "integrated.rpca",
  verbose = FALSE
)
```
~~~